Online Recommendation Engine Using Web Scraping Data

```
#Loading the required libraries
library(rvest)
library(ggplot2)
library(rsconnect)
library(shiny)

#Specifying the URL for the website to be scraped and I have used IMDB Top 100 feature films in
 2016
url <- 'http://www.imdb.com/search/title?count=100&release_date=2016,2016&title_type=feature'

#Reading the HTML code from the website
webpage <- read_html(url)

#Using CSS selectors to scrap the rankings section
rank_data_html <- html_nodes(webpage,'.text-primary')

#Converting the ranking data to text
rank_data <- html_text(rank_data_html)

#Let's have a look at the rankings
head(rank_data)
```

```
## [1] "1." "2." "3." "4." "5." "6."
```

```
#Checking the data type of rank_data
class(rank_data)
```

```
## [1] "character"
```

```
#Data-Preprocessing: Converting rankings to numerical
rank_data<-as.numeric(rank_data)

#Let's have another look at the rankings
head(rank_data)
```

```
## [1] 1 2 3 4 5 6
```

```
#Using CSS selectors to scrap the title section
title_data_html <- html_nodes(webpage,'.lister-item-header a')

#Converting the title data to text
title_data <- html_text(title_data_html)

#Let's have a look at the title
head(title_data)
```

```
## [1] "Deadpool"                   "Suicide Squad"
## [3] "Doctor Strange"             "Ghostbusters"
## [5] "Captain America: Civil War" "Moana"
```

```
#Using CSS selectors to scrap the description section
description_data_html <- html_nodes(webpage,'.ratings-bar+ .text-muted')

#Converting the description data to text
description_data <- html_text(description_data_html)

#Let's have a look at the description data
head(description_data)
```

```
## [1] "\n    A fast-talking mercenary with a morbid sense of humor is subjected to a rogue expe
riment that leaves him with accelerated healing powers and a quest for revenge."

## [2] "\n    A secret government agency recruits some of the most dangerous incarcerated super-
villains to form a defensive task force. Their first mission: save the world from the apocalyps
e."
## [3] "\n    While on a journey of physical and spiritual healing, a brilliant neurosurgeon is
drawn into the world of the mystic arts."

## [4] "\n    Following a ghost invasion of Manhattan, paranormal enthusiasts Erin Gilbert and A
bby Yates, nuclear engineer Jillian Holtzmann, and subway worker Patty Tolan band together to st
op the otherworldly threat."
## [5] "\n    Political involvement in the Avengers' activities causes a rift between Captain Am
erica and Iron Man."

## [6] "\n    In Ancient Polynesia, when a terrible curse incurred by the Demigod Maui reaches M
oana's island, she answers the Ocean's call to seek out the Demigod to set things right."
```

```
#Data-Preprocessing: removing '\n'
description_data<-gsub("\n","",description_data)

#Let's have another look at the description data
head(description_data)
```

```
## [1] "    A fast-talking mercenary with a morbid sense of humor is subjected to a rogue experi
ment that leaves him with accelerated healing powers and a quest for revenge."

## [2] "    A secret government agency recruits some of the most dangerous incarcerated super-vi
llains to form a defensive task force. Their first mission: save the world from the apocalypse."

## [3] "    While on a journey of physical and spiritual healing, a brilliant neurosurgeon is dr
awn into the world of the mystic arts."

## [4] "    Following a ghost invasion of Manhattan, paranormal enthusiasts Erin Gilbert and Abb
y Yates, nuclear engineer Jillian Holtzmann, and subway worker Patty Tolan band together to stop
the otherworldly threat."
## [5] "    Political involvement in the Avengers' activities causes a rift between Captain Amer
ica and Iron Man."

## [6] "    In Ancient Polynesia, when a terrible curse incurred by the Demigod Maui reaches Moa
na's island, she answers the Ocean's call to seek out the Demigod to set things right."
```

```
#Using CSS selectors to scrap the Movie runtime section
runtime_data_html <- html_nodes(webpage,'.text-muted .runtime')

#Converting the runtime data to text
runtime_data <- html_text(runtime_data_html)

#Let's have a look at the runtime
head(runtime_data)
```

```
## [1] "108 min" "123 min" "115 min" "116 min" "147 min" "107 min"
```

```
#Data-Preprocessing: removing mins and converting it to numerical

runtime_data<-gsub(" min","",runtime_data)
runtime_data<-as.numeric(runtime_data)

#Using CSS selectors to scrap the Movie genre section
genre_data_html <- html_nodes(webpage,'.genre')

#Converting the genre data to text
genre_data <- html_text(genre_data_html)

#Let's have a look at the runtime
head(genre_data)
```

```
## [1] "\nAction, Adventure, Comedy           "
## [2] "\nAction, Adventure, Fantasy           "
## [3] "\nAction, Adventure, Fantasy           "
## [4] "\nAction, Comedy, Fantasy           "
## [5] "\nAction, Adventure, Sci-Fi           "
## [6] "\nAnimation, Adventure, Comedy           "
```

```
#Data-Preprocessing: removing \n
genre_data<-gsub("\n","",genre_data)

#Data-Preprocessing: removing excess spaces
genre_data<-gsub(" ","",genre_data)

#taking only the first genre of each movie
genre_data<-gsub(",.*","",genre_data)

#Convering each genre from text to factor
genre_data<-as.factor(genre_data)

#Let's have another look at the genre data
head(genre_data)
```

```
## [1] Action    Action    Action    Action    Action    Animation
## 8 Levels: Action Adventure Animation Biography Comedy Crime ... Horror
```

```
#Using CSS selectors to scrap the IMDB rating section
rating_data_html <- html_nodes(webpage,'.ratings-imdb-rating strong')

#Converting the ratings data to text
rating_data <- html_text(rating_data_html)

#Let's have a look at the ratings
head(rating_data)
```

```
## [1] "8.0" "6.1" "7.5" "5.3" "7.8" "7.6"
```

```
#Data-Preprocessing: converting ratings to numerical
rating_data<-as.numeric(rating_data)

#Let's have another look at the ratings data
head(rating_data)
```

```
## [1] 8.0 6.1 7.5 5.3 7.8 7.6
```

```
#Using CSS selectors to scrap the votes section
votes_data_html <- html_nodes(webpage,'.sort-num_votes-visible span:nth-child(2)')

#Converting the votes data to text
votes_data <- html_text(votes_data_html)

#Let's have a look at the votes data
head(votes_data)
```

```
## [1] "740,284" "481,012" "427,989" "170,275" "506,427" "192,649"
```

```
#Data-Preprocessing: removing commas
votes_data<-gsub(",","",votes_data)

#Data-Preprocessing: converting votes to numerical
votes_data<-as.numeric(votes_data)

#Let's have another look at the votes data
head(votes_data)
```

```
## [1] 740284 481012 427989 170275 506427 192649
```

```
#Using CSS selectors to scrap the directors section
directors_data_html <- html_nodes(webpage,'.text-muted+ p a:nth-child(1)')

#Converting the directors data to text
directors_data <- html_text(directors_data_html)

#Let's have a look at the directors data
head(directors_data)
```

```
## [1] "Tim Miller"       "David Ayer"       "Scott Derrickson"
## [4] "Paul Feig"        "Anthony Russo"    "Ron Clements"
```

```
#Data-Preprocessing: converting directors data into factors
directors_data<-as.factor(directors_data)

#Using CSS selectors to scrap the actors section
actors_data_html <- html_nodes(webpage,'.lister-item-content .ghost+ a')

#Converting the gross actors data to text
actors_data <- html_text(actors_data_html)

#Let's have a look at the actors data
head(actors_data)
```

```
## [1] "Ryan Reynolds"        "Will Smith"           "Benedict Cumberbatch"
## [4] "Melissa McCarthy"     "Chris Evans"          "Auli'i Cravalho"
```

```
#Data-Preprocessing: converting actors data into factors
actors_data<-as.factor(actors_data)

#Using CSS selectors to scrap the metascore section
metascore_data_html <- html_nodes(webpage,'.metascore')

#Converting the runtime data to text
metascore_data <- html_text(metascore_data_html)

#Let's have a look at the metascore
head(metascore_data)
```

```
## [1] "65        " "40        " "72        " "60        " "75        "
## [6] "81        "
```

```
#Data-Preprocessing: removing extra space in metascore
metascore_data<-gsub(" ","",metascore_data)

#Lets check the length of metascore data
length(metascore_data)
```

```
## [1] 97
```

```
#After visual inspection Metascore is missing for 3 movies with rankings 31,68,93
for (i in c(31,68,93)){

  a<-metascore_data[1:(i-1)]

  b<-metascore_data[i:length(metascore_data)]

  metascore_data<-append(a,list("NA"))

  metascore_data<-append(metascore_data,b)

}

#Let's have another look at length of the metascore data
length(metascore_data)
```

```
## [1] 100
```

```
#Data-Preprocessing: converting metascore to numerical
metascore_data<-as.numeric(metascore_data)
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
#Let's look at summary statistics
summary(metascore_data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    21.0    44.0    58.0    58.3    71.0    99.0       3
```

```
#Using CSS selectors to scrap the gross revenue section
gross_data_html <- html_nodes(webpage,'.ghost~ .text-muted+ span')

#Converting the gross revenue data to text
gross_data <- html_text(gross_data_html)

#Let's have a look at the votes data
head(gross_data)
```

```
## [1] "$363.07M" "$325.10M" "$232.64M" "$128.34M" "$408.08M" "$248.76M"
```

```
#Data-Preprocessing: removing '$' and 'M' signs
gross_data<-gsub("M","",gross_data)

gross_data<-substring(gross_data,2,6)

#Let's check the length of gross data
length(gross_data)
```

```
## [1] 91
```

```
#Filling missing entries of gross data with NA for below movie rankings
for (i in c(18,21,31,68,72,81,88,89,93)){

  a<-gross_data[1:(i-1)]

  b<-gross_data[i:length(gross_data)]

  gross_data<-append(a,list("NA"))

  gross_data<-append(gross_data,b)

}

#Data-Preprocessing: converting gross to numerical
gross_data<-as.numeric(gross_data)
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
#Let's have another look at the length of gross data
length(gross_data)
```

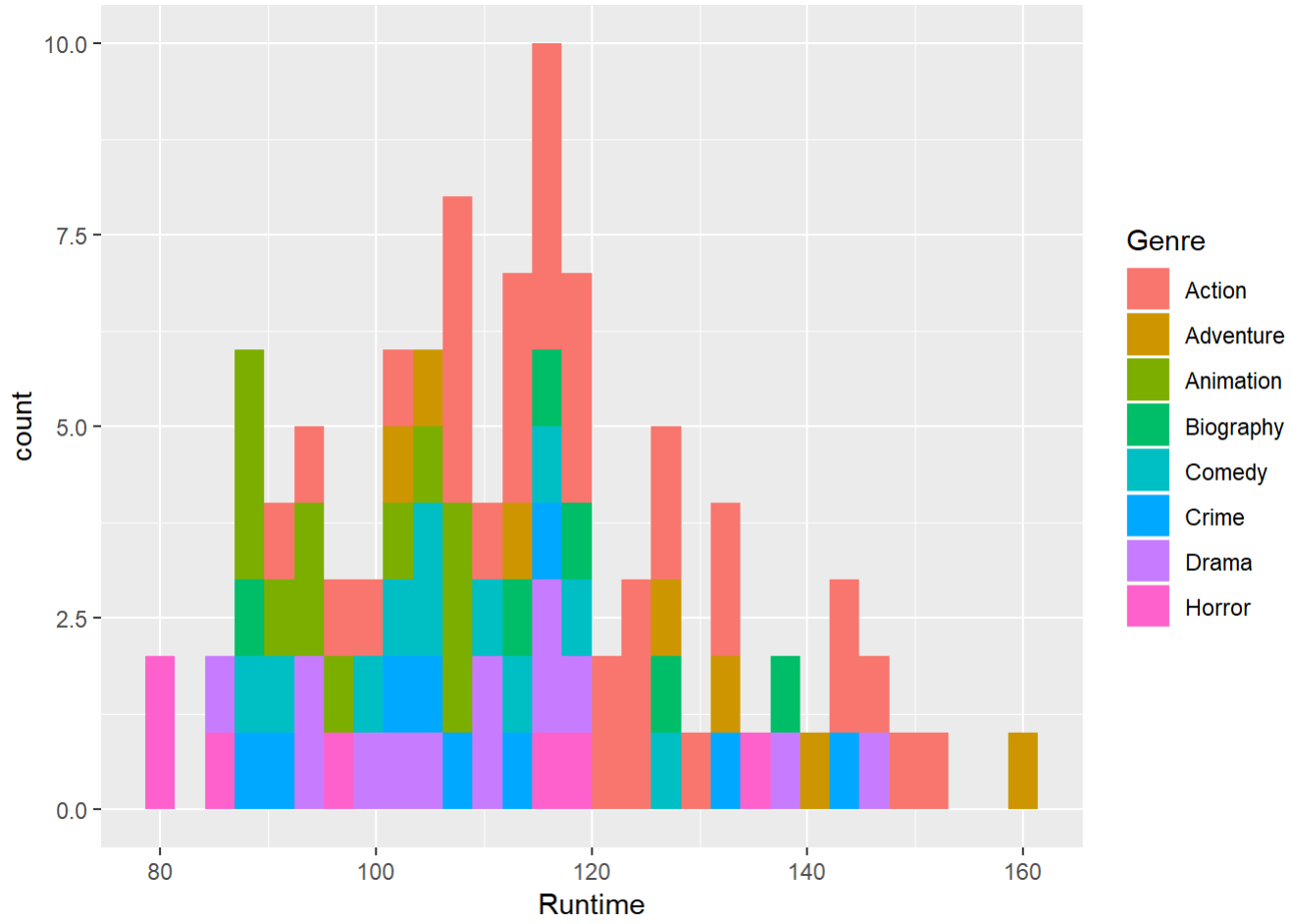
```
## [1] 100
```

```
#Combining all the lists to form a data frame
movies_df<-data.frame(Rank = rank_data[1:100], Title = title_data[1:100],

                      Description = description_data[1:100], Runtime = runtime_data[1:100],

                      Genre = genre_data[1:100], Rating = rating_data[1:100],

                      Metascore = metascore_data[1:100], Votes = votes_data[1:100],

                      Gross_Earning_in_Mil = gross_data, Director = directors_data[1:100], Actor
 = actors_data[1:100])

#Structure of the data frame
str(movies_df)
```

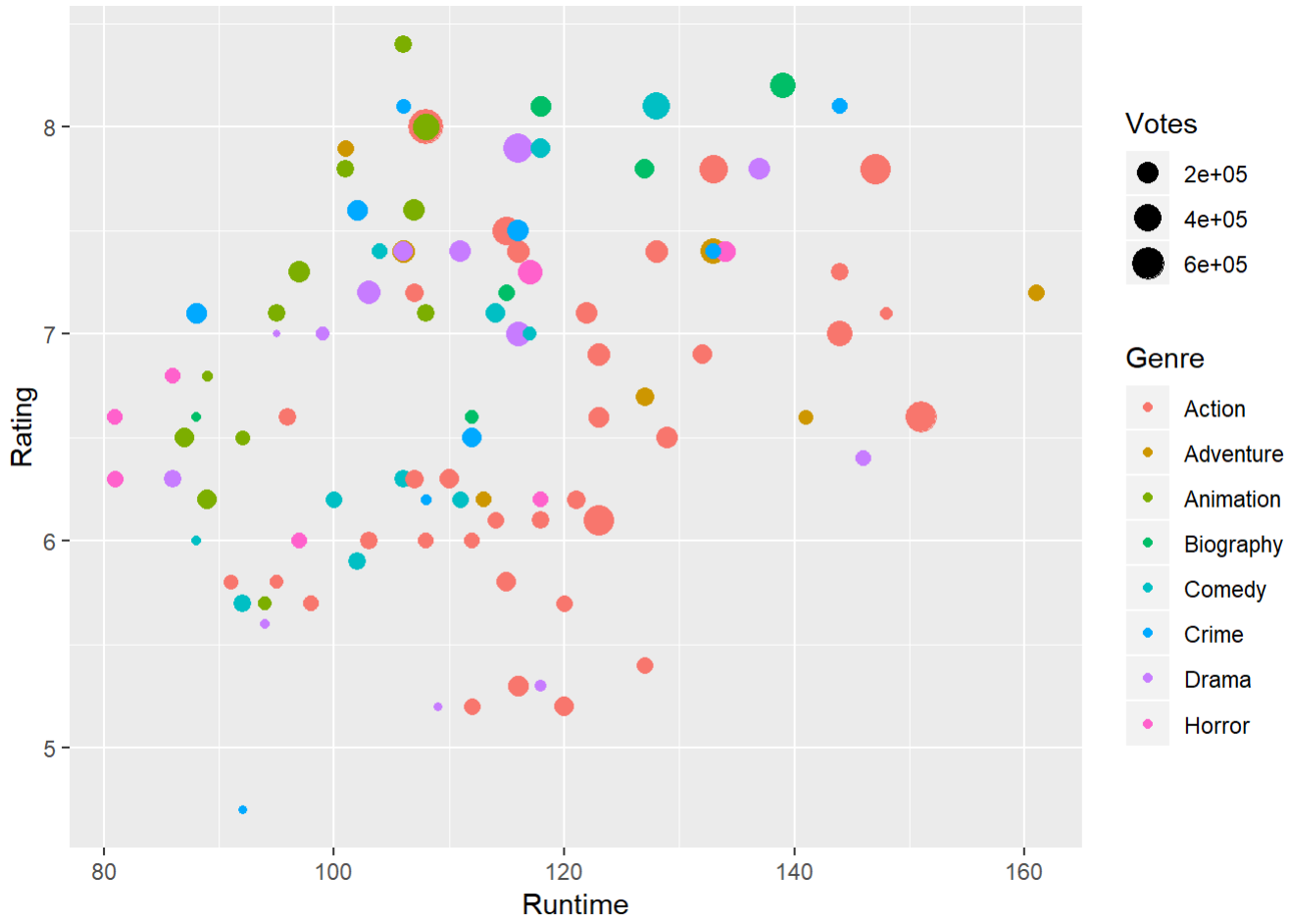
```
## 'data.frame':    100 obs. of  11 variables:
##  $ Rank               : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Title              : Factor w/ 100 levels "10 Cloverfield Lane",..: 20 70 23 28 15 54 76
75 67 11 ...
##  $ Description        : Factor w/ 100 levels "    A blind woman's relationship with her husb
and changes when she regains her sight and discovers disturbing d"| __truncated__,..: 9 23 98 54
72 65 39 64 63 53 ...
##  $ Runtime            : num  108 123 115 116 147 107 97 118 108 151 ...
##  $ Genre              : Factor w/ 8 levels "Action","Adventure",..: 1 1 1 1 1 3 8 7 3 1 ...
##  $ Rating             : num  8 6.1 7.5 5.3 7.8 7.6 6 5.3 7.1 6.6 ...
##  $ Metascore          : num  65 40 72 60 75 81 42 62 59 44 ...
##  $ Votes              : num  740284 481012 427989 170275 506427 ...
##  $ Gross_Earning_in_Mil: num  363 325 233 128 408 ...
##  $ Director           : Factor w/ 98 levels "Alessandro Carloni",..: 91 21 83 73 8 81 96 4 3
4 98 ...
##  $ Actor              : Factor w/ 89 levels "Alexander Skarsgård",..: 72 87 7 61 17 5 51 78
59 6 ...
```

```
#Data which got from web-scraping
head(movies_df)
```

```
#Analyzing which movie from which Genre has the highest and Lowest Run time
ggplot(data=movies_df,aes(Runtime))+geom_histogram(aes(fill=Genre),bins=30)
```

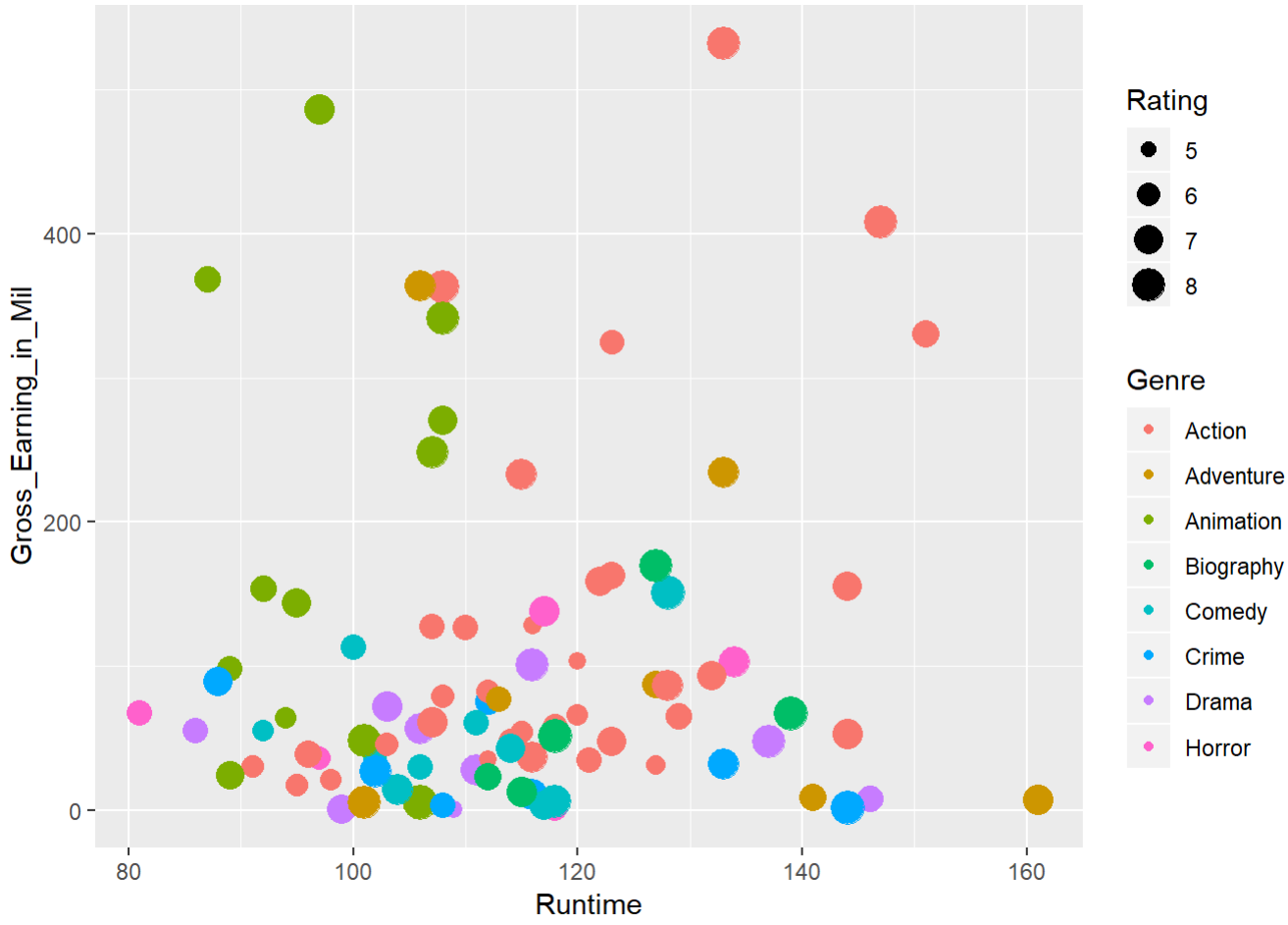


```
#Analyzing which Genre has the highest votes with different Run times
ggplot(data=movies_df,aes(Runtime,Rating))+geom_point(aes(size=Votes,color=Genre))
```




```
#Analyzing across all Genres which has highest average gross earnings in runtime 100 to 120
ggplot(data=movies_df,aes(Runtime,Gross_Earning_in_Mil))+geom_point(aes(size=Rating,color=Genr
e))
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```


Gross_Earning_in_Mil
400
200
0
80
100
120
140
160
Runtime
Rating
5
6
7
8
Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Horror

```
#ui.R
    ui <- fluidPage(
      titlePanel("Online Movie Recommendation Engine"),
        fluidRow(
          column(5,
              selectInput(inputId = "movie1", label = h3("Choose Three Movies You Like"),
                       choices = as.character(movies_df$Title)),

              selectInput(inputId = "movie2", label = NA,
                       choices = as.character(movies_df$Title)),

              selectInput(inputId = "movie3", label = NA,
                       choices = as.character(movies_df$Title)),

              submitButton("Submit")
       ),

    column(7,
           h3("You Might Like These Too!"),
           tableOutput("table"))
  )
)

#server.R
server<- function(input,output){
  output$table <- renderTable({
      #Getting the user inputted movie values from ui.R
      movie1 <- input$movie1
      movie2 <- input$movie2
      movie3 <- input$movie3

      #Reading the csv file created in ui.R
      #movies_dfs<-read.csv("IMDBmovies.csv")

      #Creating empty vector for recommended movies to the user
      recommendedmovies<-c()

      #Combining the input movies in to a vector
      mymovies<-c()
      mymovies<-c(mymovies,movie1,movie2,movie3)

      #Finding recommended movies using Genre of user inputted movies
      ##################Context Based Filtering Approach###################

      for (eachmovie in mymovies)
      {
        eachgenre<-movies_df$Genre[movies_df$Title==eachmovie]
        allmoviesgenre<-movies_df$Title[movies_df$Genre==as.character(eachgenre) & movies_df$Tit
le!=eachmovie]
        recommendedmovies<-c(recommendedmovies,head(as.character(allmoviesgenre),3))
      }

      #Converting in to dataframe
```

```
      recommendedmovies_df<-as.data.frame(recommendedmovies)

      recommendedmovies_df
    })
}

# Run the app
shinyApp(ui = ui, server = server)
```

Online Movie Recommendation Engine

Choose Three Movies You Like

Captain America: Civil War

Ice Age: Collision Course

Hail, Caesar!

Submit

You Might Like These Too!

recommendedmovies